Modélisation et analyses d’architecture IMA à l’aide d’AADL : analyses de sûreté de fonctionnement et génération de code.


To cite this version : Hugues, Jérôme and Delange, Julien
Modélisation et analyses d’architecture IMA à l’aide d’AADL : analyses de sûreté de fonctionnement et génération de code.

Any correspondance concerning this service should be sent to the repository administrator: staff-oatao@listes-diff.inp-toulouse.fr
Model-Based Design, Automated Code Generation and Safety Analysis of ARINC653 Architectures using the AADL

Jérôme Hugues, ISAE
Julien Delange, CMU/SEI
One fault instance of an ADIRU (Air Data Inertial Reference Unit) on-board a Boeing 777-2H6ER caused a hazardous accident to Malaysian Air flight 124 in 2005,

Key question is: could we avoid similar scenario in future system design? How? Associated cost?

Failure has been (partially) described in publicly available reports by NTSB, and Vanderbilt University, used for study.

---

**Agenda**

1. How to capture architecture key elements using AADL
   - Real-time architecture, ARINC653 patterns, etc.
2. Link them to implementation artifacts
   - Simulation through code generation
3. Trace them w.r.t. safety analysis objectives
Multiple levels of redundancy.
- work without maintenance with one fault in each FCA.
> ISIS-11-101 TR by Vanderbilt Univ.
• Four modules
• Two types of ports
1. Capturing architecture key elements using AADL
   • Real-time architecture, ARINC653 patterns, etc.

2. Link them to implementation artifacts
   • Simulation through code generation

3. Trace them w.r.t. safety analysis objectives
International standard promoted by SAE, AS-2C committee
» Released as AS5506 family of standards
» Based on feedback from the aerospace industry

Annex document to address specific needs
» ARINC653, Behavior, data, error modeling, code
generation, …

AADL objectives are “to model a system”
» With analysis in mind
» To ease transition from well-defined requirements
to the final system: code production

Require semantics => any AADL entity has a semantics
(natural language or formal methods).
Regular modeling process

» Define sub-system boundaries, interfaces, configuration

» Mixing text, graphics, property editor to manage model complexity
Overview of the AADL model
AADL default semantics check

- Containment hierarchy, applicability of configuration parameters (units, types, etc), types of message exchanged, port connection, etc.

ARINC 653 verification plugs-ins

- Part of AADL eco-system: OSATE, MASIW, Ocarina, …
- Check connections
- Validity of ARINC653 Configuration parameters:
  - Major Frame Correctness, Properties of Memory Components, Dimensioning of Memory Components, Partitions Bindings, Partitions Executions, Separation of Memory
- Additional checks: constraints set by RTOS vendors, e.g. alignment of memory segments, max number of threads, ports, size of queues, etc.
1. Capturing architecture key elements using AADL
   • Real-time architecture, ARINC653 patterns, etc.
2. Link them to implementation artifacts
   • Simulation through code generation
3. Trace them w.r.t. safety analysis objectives
ARINC653 Executives require an additional configuration file, but …

A (full) AADL model must define all components
» For analysis or code generation purposes

Can derive configuration file from the AADL model
» Implemented in Ocarina, targets DeOS and VxWork653

Part of the model bus philosophy
» One repository that can be mined for various purposes
» Analysis, code generation, management of configuration parameters
-- Part of the Annex D - Data Modeling Annex

data C_Unsigned_Long_Int
   -- This data component defines a C unsigned long int type, with a
   -- dual nature The first properties defines its representation in
   -- memory, the two last its mapping in C.
properties
   Data_Model::Data_Representation => integer;
   Data_Model::Number_Representation => unsigned;
   Data_Size => 4 bytes;
   Source_Language => (C);
   Type_Source_Name => "unsigned long int";
end C_Unsigned_Long_Int;

data accData extends C_Unsigned_Long_Int
end accData;

subprogram acl1_dataOutput_spg
features
   acl1DataOut: out parameter SHM_DataType::accData;
   event_in: in parameter SHM_DataType::actionData;
end acl1_dataOutput_spg;
AADL and subprograms

> Binding code to AADL components

```plaintext
subprogram accl_dataOutput_spg
features
  acc1DataOut: out parameter SHM_DataType::accData;
  event_in: in parameter SHM_DataType::actionData;
properties
  Source_Language => (C);
  Source_Name => "accldataoutput";
  Source_Text => ("../../../acc_code.o");
end accl_dataOutput_spg;
```

> Mapping from AADL model to code

```plaintext
subprogram accl_dataOutput_spg
features
  acc1DataOut: out parameter SHM_DataType::accData;
  event_in: in parameter SHM_DataType::actionData;
end accl_dataOutput_spg;

void accl_dataOutput_spg ( /* C */
  (acc1DataOut *SHM_DataType_accData,
   event_in: SHM_DataType_actionData);
```
The AADL architecture has all details about
» Task, queues, buffers, etc.
» Used for schedulability analysis, generation of ARINC653 configuration

Ocarina: massive code generation
» Take advantage of global knowledge to optimize code, and generate only what is required
» Reduce as much as possible error-prone and tedious tasks

Targets DeOS and VxWorks 653
• See all demos and videos from http://aadl.info/aadl/demo-arinc653/
1. Capturing architecture key elements using AADL
   • Real-time architecture, ARINC653 patterns, etc.
2. Link them to implementation artifacts
   • Simulation through code generation
3. Trace them w.r.t. safety analysis objectives
System safety process uses many individual methods and analyses, e.g.
- hazard analysis
- failure modes and effects analysis
- fault trees
- Markov processes

Related analyses are also useful for other purposes, e.g.
- maintainability
- availability
- Integrity

Goal: a general facility for modeling fault/error/failure behaviors that can be used for several modeling and analysis activities.

SAE ARP 4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment
Automation of SAE ARP4761 System Safety Assessment Practice

FHA
Spreadsheet
Uses error sources

FMEAs
Spreadsheet
Uses error flows & propagations

FTA
CAFTA, OpenFTA
Uses composite error behavior

Markov Chain
PRISM
Uses error flows & behavior

RBD/DD
OSATE plugin
Uses composite error behavior

AADL & EMV2

Uses composite error behavior
Annotating the model with Error Information (1)

Declaring error sources

Documenting the error

```plaintext
device implementation acc_device.impl
annex EMV2
{**
  use types ADIRU_errLibrary;
  use behavior ADIRU_errLibrary::simple;

  error propagations
  accData : out propagation{ValueErroneous};
  flows
    f1 : error source accData{ValueErroneous} when failed;
  end propagations;

  properties
  emv2::hazard =
  ( [ crossreference => "N/A";
      failure  => "Accelerometer value error";
      phases  => ("in flight");
      description => "Accelerometer starts to send an erroneous value";
      comment => "Can be critical if not detected by the health monitoring";
    ])
  applies to accData.valueerroneous;

  EMV2::OccurrenceDistribution => { ProbabilityValue => 3.4e-5 ; Distribution => Fixed;}
  applies to accData.valueerroneous;

  **};
end acc_device.impl;
```
Annotating the model with Error Information (2)

Passing the error directly through components features
Receiving a erroneous value makes the component to fail.
Functional Hazard Assessment:
• List all potential error sources, include documentation from the model

<table>
<thead>
<tr>
<th>Component</th>
<th>Error</th>
<th>Hazard Description</th>
<th>osse referred</th>
<th>Functional Failure</th>
<th>Operational Phases</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>acc1</td>
<td>&quot;ValueErroneous on accData&quot;</td>
<td>&quot;Accelerometer starts to send an erroneous value&quot;</td>
<td>&quot;N/A&quot;</td>
<td>&quot;Accelerometer value error&quot;</td>
<td>&quot;in flight&quot;</td>
<td>&quot;Can be critical if not detected by the health monitoring&quot;</td>
</tr>
<tr>
<td>acc2</td>
<td>&quot;ValueErroneous on accData&quot;</td>
<td>&quot;Accelerometer starts to send an erroneous value&quot;</td>
<td>&quot;N/A&quot;</td>
<td>&quot;Accelerometer value error&quot;</td>
<td>&quot;in flight&quot;</td>
<td>&quot;Can be critical if not detected by the health monitoring&quot;</td>
</tr>
<tr>
<td>acc3</td>
<td>&quot;ValueErroneous on accData&quot;</td>
<td>&quot;Accelerometer starts to send an erroneous value&quot;</td>
<td>&quot;N/A&quot;</td>
<td>&quot;Accelerometer value error&quot;</td>
<td>&quot;in flight&quot;</td>
<td>&quot;Can be critical if not detected by the health monitoring&quot;</td>
</tr>
<tr>
<td>acc4</td>
<td>&quot;ValueErroneous on accData&quot;</td>
<td>&quot;Accelerometer starts to send an erroneous value&quot;</td>
<td>&quot;N/A&quot;</td>
<td>&quot;Accelerometer value error&quot;</td>
<td>&quot;in flight&quot;</td>
<td>&quot;Can be critical if not detected by the health monitoring&quot;</td>
</tr>
<tr>
<td>acc5</td>
<td>&quot;ValueErroneous on accData&quot;</td>
<td>&quot;Accelerometer starts to send an erroneous value&quot;</td>
<td>&quot;N/A&quot;</td>
<td>&quot;Accelerometer value error&quot;</td>
<td>&quot;in flight&quot;</td>
<td>&quot;Can be critical if not detected by the health monitoring&quot;</td>
</tr>
<tr>
<td>acc6</td>
<td>&quot;ValueErroneous on accData&quot;</td>
<td>&quot;Accelerometer starts to send an erroneous value&quot;</td>
<td>&quot;N/A&quot;</td>
<td>&quot;Accelerometer value error&quot;</td>
<td>&quot;in flight&quot;</td>
<td>&quot;Can be critical if not detected by the health monitoring&quot;</td>
</tr>
</tbody>
</table>

Fault Impact Analysis
• Bottom-up approach, trace the error flow defined in the architecture

Fault Tree

> Fault Tree
> AADLv2 leveraged to model the ADIRU system
  » Full architectural description of the avionics system
  » Link with consistency checks for ARINC653 patterns
  » Code generation towards ARINC653 APEX
  » Safety analysis using the AADL EMV2 annex

> AADL ecosystem provide all required tools, using OSATE2 and Ocarina, completed with spreadsheets, FTA tool and target RTOS

> Future work will consider connection with requirement engineering, and better coverage of faulty scenarios